Scalable and Robust Adversarial Attack against Deep Reinforcement Learning

***Abstract*—Abstract goes here. convert the draft to AAAI format**

Backdoor attacks in Deep Neural Networks (DNN) have always been studied and various defenses have been proposed to detect and mitigate these attacks. However, backdoor attacks in Deep Reinforecement Learning (DRL) setting is largely unexplored. The high performance of DRL algorithms in complex tasks increases the need of robust DRL. We need to develop efficient attacks for robust DRL. In this paper, we introduce a novel attack called fast defender aware attack. It uses existing scalable technique of creating adversarial examples in DNN setting to compromise the end goal of DRL agent. Previously, fast and scalable methods to generate perturbations used in DRL-based attacks have been easily detected by state of the art defenses such as adversarial retraining. We show that our attack is comparatively robust against such defenses. This attack also devised a novel defender aware strategy to choose the steps to attack in an episode. We also compare this technique with other state of the art attacks to show that our attack is fast and robust with comparable success rate. We also introduced a novel metric, defender aware success rate, to evaluate our attack.

I. Introduction

A. Motivation

answer the following: (1) why this problem is important; (2) why this problem is challenging; (3) what has been done in the existing works (very briefly) and what are their limitations.

Deep Reinforcement Learning (DRL) algorithms learn policies to guide DRL agent to take optimal actions based on environment state. These algorithms achieved really high performance on various complex as well as critical tasks such as robotics Amarjyoti [2], autonomous driving Kiran et al. [8] etc. This policy is often learnt by Deep Neural Networks (DNN) for the approximation of action-value function.

DNN are vulnerable to adversarial attacks Goodfellow et al. [5], Szegedy et al. [21]. One example attack model is to add small perturbations in the images, which can mislead a Convolutional Neural Network (CNN) to give adversary's output. Various attacks and defenses have been proposed for supervised DNN applications such as image classification Goodfellow et al. [5] and natural language processing Alzantot et al. [1]. However, adversarial attacks and defenses are largely unexplored in DRL-based setting. DRL has numerous critical applications in safety and security, which highlights the importance of robust DRL. For robust DRL, there is a pre-requisite to develop efficient adversarial attacks.

We divide an adversarial attack into two major questions in DRL based setting. 1) *How to Attack?* 2) *When to Attack?*. However, in DNN based setting, there is majorly one question of *How to Attack?*. Compromising the end goal of DRL agent is more important than mere reducing the reward in few episodes. To compromise the end goal, we also need to develop perturbations in a scalable way and simultaneously make sure that our perturbations are strong and accurate enough to mislead the DRL agent. This makes developing robust DRL attacks challenging. When the attacker compromises the end goal of DRL agent in an aggressive way that an anomalous behavior is visible to defender, then attack can be identified by the defender and attacker's attack technique may not work later. Our attack considers the normal and anomalous actions in different states to remain as unidentifiable as possible to the defender, while still compromising the end goal of the DRL agent. In summary, to conduct a robust and efficient attack, the goal of our attacker is to add small, scalable, human-imperceptible and successful perturbations in a minimum number of steps during an episode to compromise the end goal of DRL agent, while remaining undetectable.

To address this goal, there are some efforts to develop adversarial attacks against DRL. Huang et al. [7] made a preliminary attempt to attack the DRL agent in each step using fast gradient sign method (FGSM) Goodfellow et al. [6]. However, this attempt only considered the naive approach of FGSM and did not consider compromising the end goal of DRL agent. The drawback of naive FGSM is that it has not 100% success rate and is easily detectable. Lin et al. [12], Sun et al. [20] considered developing sophisticated attacks based on end goal strategies but their perturbation generation method is based on Carlini and Wagner [4]. Carlini and Wagner [4] methods are 10-100X slower than naive FGSM based perturbations since they achieved 100% success rate. Hence, there is a need of developing a less detectable and fast attack strategy with reasonably good success rate in DRL based settings where attacking in a less time is the key. Such attacks have been studied in DNN based settings Tramèr et al. [22] but there is still a need to develop such attacks in DRL based setting. Previous attacks are easily indentifiable by the defenders as they make the DRL agent completely anomalous. In DRL attacks, there is no attack which developed defender aware strategy and considered undetectability as their key property. Our work, first of its kind, considers developing defender-aware, long term and fast attack.

In this paper, we propose a robust and fast attack with comparable success rate while compromising the end goal of a DRL agent in an undetectable way. We call this attack: **Fast Defender Aware Attack**. This attack addresses the question of *How to Attack?* by developing more sophisticated and robust FGSM based methods in DRL based setting. For *When to Attack?*, we find implicit attack strategies and remove explicit attack strategies to achieve severe damage in an undetectable way.

B. Research Goal and Questions

The overarching goal of this work: We aim to....; describe the problem statement in a high level

We have the following **research goals and questions** to be addressed in this work:

1) We aim to make more robust attack by generating more robust adversarial examples in the Deep Reinforcement Learning setting.
2) We aim to perform attack in minimal number of steps at critical stages during the episode, resulting in huge negative reward.
3) While reducing reward and attacking end goal, we aim to remain undetectable by the defender.
4) We aim to evaluate this attack using state of the art DRL defenses such as adversarial retraining and compare with the baseline FGSM based attacks.
5) We want to perform this attack on DQN and A3C methods in white box setting using Atari Games.

C. Key Contributions

The **key contributions** of this paper are:

1) Design of Fast defender aware attack which can generate fast and robust perturbations while developing a defender aware strategy to attack sequence of critical steps to compromise the end goal of DRL agent.
2) We gave a new metric called defender aware success rate to evaluate our attack.
3) Comprehensive Evaluation against state of the art attacks and defenses

D. Structure of the Paper

This can be done after completing all sections

II. Related Work .5 page

A. Backdoor Attacks in Deep Neural Networks (supervised DL)

(1) discuss the state-of-the-art backdoor attacks developed to disturb DNN; (2) discuss whether they are also considered in DRL context; (3) discuss limitations in terms of not much being used in DRL context

Szegedy et al. [21] initially presented an attack that small perturbations can lead to misclassifications. Later, various attacks and defenses have been proposed. Some attacks were fast and had reasonable success rate such as fast gradient sign method based attacks Goodfellow et al. [5] and randomness based fast gradient sign method attacks Tramèr et al. [22]. Carlini and Wagner [4] attacks gave 100% success rate but their perturbation generation method is very slow. None of these studies considered attacking on Deep Reinforcement Learning. The perturbation methods of Carlini and Wagner [4], Goodfellow et al. [5] have been used in DRL context but the concept of randomness, robust yet fast perturbation methods given by Tramèr et al. [22] have not been used in DRL context.

B. Adversarial Attacks in Deep Reinforcement Learning

Huang et al. [7] made a preliminary attempt to attack the DRL agent in each step using fast gradient sign method (FGSM) Goodfellow et al. [6]. However, this attempt only considered the naive approach of FGSM and did not consider compromising the end goal of DRL agent. The drawback of naive FGSM is that it has not 100% success rate and is easily detectable. Lin et al. [12], Sun et al. [20] considered developing sophisticated attacks based on end goal strategies but their perturbation generation method is based on Carlini and Wagner [4]. Carlini and Wagner [4] methods are 10-100X slower than naive FGSM based perturbations since they achieved 100% success rate. Hence, there is a need of developing a less detectable and fast attack strategy with reasonably good success rate in DRL based settings where attacking in a less time is the key. Such attacks have been studied in DNN based settings Tramèr et al. [22] but there is still a need to develop such attacks in DRL based setting.

Sun et al. [20] considered end goal strategy in their Critical Point Attack and Antagonist Attack, and attacked in a minimum number of steps. But there are two problems in their methodology: 1- Their attacks consider all possibilities to find the worst action, which is time consuming. 2- They targeted the end goal of the DRL agent and chosen the worst action as the adversary's goal, which can be easily identified by a defender that there is a problem with DRL agent. Our goal is not only devise a strategy that is fast but also that is hidden enough not to be identifiable.

(1) discuss popular attacks considered in DRL and discuss backdoor attacks addressed in DRL if any; emphasize there is not much work investigating particularly backdoor attacks in DNN side; and (2) add discussions on how our approach differs from each of the existing counterparts; can make use of a table summarizing what types of attacks are considered and what types of defense mechanisms are considered

III. Preliminaries

A. Deep Reinforcement Learning

Reinforcement learning algorithms optimize the expected cumulative reward by training a policy π. This policy can be deterministic or probabilistic function that maps each state to an action

$$\pi : S \to A$$

where S and A are state and action space respectively. In deep reinforcement learning (DRL), this function is learned by a neural network. Deep Q-networks (DQN) [14], Asynchronous Advantage Actor-Critic (A3C) [16] and Trust Region Policy Optimization (TRPO) [19] are some of the famous DRL algorithms. We have majorly evaluated our attack on the policy trained by DQN.

1) Deep Q-Networks (DQN): In Q-learning, the Q-value is the expected cumulative discounted reward when an action a is taken in state s. DQN is a kind of Q-learning where Q-values are learned by a neural network while minimizing the squared Bellman error. Experience Replay helps decrease the high variance due to Q-learning updates. A replay buffer stores all recent transitions. To mitigate the correlation due to time, random samples are taken from this buffer. DQN takes an action based on the maximum Q-value.

2) Advantage Actor-Critic (A3C): A3C has two components: an actor which is a policy based on neural network and a critic which is based on a value function. In A3C, the learning of a stochastic policy is stabilized and boosted up by an asynchronous gradient descent. The policy is updated by small batches during its learning and samples correlation is decreased by asynchronous gradient descent.

B. Fast Adversarial Example Generation Methods

1) Fast Gradient Sign Method (FGSM): It is a fast technique [5] to design adversarial examples rather than optimal adversarial examples. For an image x, a perturbed image x' according to FGSM is:

$$x'_{FGSM} = x + \epsilon \cdot sign(\nabla loss(h(x), y_{true}))$$

where t is the target label and ϵ is the parameter to make the perturbation small enough to be less noticeable. Loss function can be a cross entropy function. FGSM do the linear approximation of the model and solves the maximization problem in closed form, which makes this method very fast. There are three kinds of norm constraints L_0, L_1 and L_∞ which are used in literature to constrain the perturbation in order not to be detectable. For n be zero, one or infinite norm, constraint can be seen in following inequality:

$$||x - x'||_n < \epsilon$$

We always need computationally efficient techniques of generating perturbations even if they give less success rate. This is also one of the goals of this paper to make a robust, undetectable and efficient technique with reasonable success rate in DRL based setting.

2) Single-Step Least-Likely Class Method (Step-LL): Kurakin et al. [10] presented a single step method like FGSM. In FGSM, loss function was based on target label. However, in Step-LL, least-likely class is used to compute the loss function.

$$x'_{LL} = x - \epsilon \cdot sign(\nabla loss(h(x), y_{LL}))$$

3) Iterative Gradient Sign (I-FGSM): Kurakin et al. [11] presented iterative gradient sign, which is a variant of FGSM. This method takes multiple small steps of size α in the direction of the sign of gradient. However, FGSM took only one step of size ϵ. This method also clips the result of multiple steps by ϵ. It outperformed FGSM. Mathematically,

Start with $x'_0 = 0$ On every iteration:

$$x'_i = x'_{i-1} - clip_\epsilon(\alpha \cdot sign(\nabla loss_{F,t}(x'_{i-1})))$$

4) Randomized FGSM: Fast single-step methods converges to a degenerate global minimum, wherein small curvature artifacts near the data points obfuscate a linear approximation of the loss. The model thus learns to generate weak perturbations, rather than defend against strong ones. To tackle this problem, Tramèr et al. [22] presented R+FGSM, which adds a small random step to such single step attacks, in order to escape the non-smooth vicinity of the data point before linearizing the model's loss. Instead of using iterative methods, this single step randomized method is computationally efficient and have high success rate.

$$x^{adv}_{R+FGSM} = x' + (\epsilon - \alpha) \cdot sign(\nabla_{x'} loss(x', y_{true}))$$

where

$$x' = x + \alpha \cdot sign(\mathcal{N}(0^d, I^d))$$

Similarly, we can add random step in Step-LL to make R+Step-LL.

C. C&W Adversarial Example Generation Technique

Carlini and Wagner [4] presented three attacks that had 100% success rate but they are 10-100X slower than naive FGSM based perturbations. We would not go into the details of their method but they were optimizing an objective function in a multi step process to generate minimal and optimal perturbations. In pursuit of this, their perturbation generation method is very slow. For exploratory research purposes and to scale the perturbation generation process, we need fast yet robust methods to generate perturbations, which is also the goal of this paper.

D. Strategy Attacks

There are a number of attacks which focused majorly on a question of *When to attack?* and ignored the time taken by their perturbation method. These attacks are efficient as they require few steps to successfully attack the DRL agent. But their runtime of generating perturbations is high as they use C&W perturbations. Our goal is to make such strategy attacks fast with reasonable success. Summaries of some of those attacks are the following:

1) Strategically Timed Attack: Lin et al. [12] presented strategically timed attack in which the adversary aims at minimizing the agent's reward by only attacking the agent at a small subset of critical time steps in an episode. However, this technique of strategically-timed attack only considered the attack effect on one step, while ignoring the impact on the following states and actions.

2) Critical Point Attack: Sun et al. [20] presented critical point attack in which the adversary builds a model to predict the future environmental states and agent's actions, assesses the damage of each possible attack strategy, and selects the optimal one. This attack requires the knowledge of domain to assess the damage.

3) Antagonist Attack: Sun et al. [20] presented antagonist attack in which the adversary automatically learns a domain agnostic model to discover the critical moments of attacking the agent in an episode.

IV. Problem Statement

clarify the problem you aim to solve; add a figure to describe the problem at a high level

DRL agent interacts with an environment and learns a policy π to choose an action *A* given a state *S*. This policy can be a probabilistic model π(s,a) $[0, 1]$, which gives the probability of taking an action given a state. It can also be a deterministic model where we can get an action directly from the policy function given the state $a = \pi$(s). The goal of DRL agent is to maximize the cumulative reward R_o by learning a policy π^*. The reward that DRL agent tries to maximize is the expected discounted reward till the next $T - 1$ time steps.

$$R_o = \sum_{t=0}^{T-1} E_{a_t \sim u(s_t)}[\gamma^t r(s_t, a_t)] \tag{1}$$

Instead of maximizing this reward, adversary has the goal to reduce the reward R_o by adding perturbation δ_t into the agent's observation s_t to mislead the agent to take the adverse action a_t. Adversary has to generate perturbation δ as small as possible in order not to be detected. Out of all time steps, adversary may or may not choose to add perturbation δ to the state s_t based on his strategy. In this way, adversary's expected cumulative reward is given by:

$$R_{adv} = \sum_{t=0}^{T-1} E_{a_t \ u(s_t + x_t \delta_t)}[\gamma^t r(s_t, a_t)]$$

where x_t at a given time t is 0 if adversary do not inject perturbation and it is 1 if it injects perturbation. We constrain the total number of time steps to be attacked by N, which should be way less than the episode length in order not to be detectable by the defender. This is given by the following inequality:

$$\sum_{t=0}^{T-1} x_t < N < \text{Episode Length}$$

The attack can be divided into two major components:

- Devise a fast, robust and undetectable strategy to find when to attack and when not to attack. It is given by a following boolean vector: $\{x_0, x_1, ..., x_{T-1}\}$
- Devise a fast and robust perturbation generation method to find a suitable δ_t whenever $x_t = 1$.

V. Proposed Approach

The proposed approach of our attack is shown in Figure 1. We consider a white box setting for our attack where adversary does not have access to the training time. However, adversary has the test time access of the DRL agent where he knows the network architecture and has access to craft his own adversarial state. The two major components of our attack are mentioned in following subsections:

1) How to Attack?: Since our primary purpose is to develop scalable, fast and robust perturbation generation methods. We are using robust gradient based methods, which were previously used to develop attacks in DNN based setting. We are extending R+FGSM and R+Step-LL methods to DRL setting. For extension, in the following equation, we consider y_{true} as the vector of weights for each action produced by the policy of DRL agent.

$$x^{adv}_{R+FGSM} = x' + (\epsilon - \alpha) \cdot sign(\nabla_{x'} loss(x', y_{true}))$$

where

$$x' = x + \alpha \cdot sign(\mathcal{N}(0^d, I^d))$$

This equation shows that we are not only moving the policy away from optimal action by using the direction of the gradient but also adding a random step in order to create strong and robust perturbations. We also tested for adding noise and uncertainties in gradient based methods to make them undetectable, as one of our goals is to make undetectable fast methods. This How to Attack? question makes our work novel as fast and robust methods have never been used in DRL based setting.

2) When to Attack?: We want to employ the strategy which has the following properties:

- It should be fast enough to decide whether to attack on current time stamp or not.
- It should compromise the end goal of DRL agent.
- It should be a hidden attack, not identifiable by the defender.

The key idea of our attack is inspired by the human decision making process. Whenever humans have certain actions and states, they evaluate all of the situations and scenarios to take an action. Sometimes, humans are in a situation where taking a rapid decision is the key. Similarly, our adversary will also evaluate the state of the DRL agent by evaluating the combinations of state action pairs. Evaluating all the state action pairs in an environment where there are high number of states and actions would be very slow for an adversary who wants to craft a rapid attack. This raises the need of an optimization problem to conduct a rapid yet optimal attack.

The solution to this optimization problem lies in the concept of hidden attack. Let us first discuss what is an hidden attack. It is an attack which is attacking in a way not visible to the defender. In order not to be identifiable, we will do following measures:

- We will try to attack on as low number of steps as possible during an episode.
- Instead of taking the worst and abnormal actions, we will try to compromise the end goal with less worse but normal actions. Because defender will easily identify the presence of adversary if an abnormal behavior will occur. We want to compromise the end goal while staying in the normal set of actions. In case of self-driving car, taking a sharp right turn on a straight road is an abnormal action and a sharp right turn on an intersection might not be considered as an abnormal action.

To make the evaluation fast and rapid, we will reduce the combinations by dropping some abnormal actions. We will need to devise a method to find abnormal actions in a certain state.

We will do the following steps to devise the attack strategy.

- Finding the normal actions in a current state. This raises the question of finding abnormal actions? Domain agnostic abnormal actions can be the those which can lead to huge negative reward. Domain aware abnormal actions can be the worst actions in that domain. Removing these abnormal actions will also make our attack fast as we will have less actions to decide our strategy from.
- Evaluation of normal combinations and finding the damage. This raises the question of how to evaluate in a less time and how to define the damage metric. For all combination of strategies, we will build a deterministic or probabilistic prediction model to predict the future states. We will find the damage metric for all these strategies we will get. We are choosing a domain agnostic metric based on the assumption that if the reward will be lower, there will be more damage. To minimize the number of evaluating combinations, we will use some greedy algorithm to make the process faster. Greedy algorithm will also help us choose the local optimal strategy rapidly.
- If the damage is greater than a threshold, we will add perturbation on the current step. This raises the question whether the action we will take now be abnormal or not. To cater this issue, we will also set an upper limit on the damage in order to take a normal yet damageable action.
- Past Experience or Utility: Adversary will learn from experience that how to attack an undetectable and robust attack. We are adding a memory component to save the transitions (current state, action, future state, reward, adversary's strategy). Adversary will learn from this memory to craft a successful attack.
-

Algorithm 1 Proposed Attack

```
Input: $M, N, s_t$
Output: optimal attack strategy { $a'_t, ...a'_{t+N-1}$ } and boolean attack variable

// Calculate the baseline $s_{t+M}$

for i in $\{0, ..., M-1\}$ do
    $a_{t+i}$ = DRLAgent($s_{t+i}$)
    $s_{t+i+1}$, reward = EnvironmentStep($s_{t+i}, a_{t+i}$)
    $s_{t+i} = s_{t+i+1}$
end for

// Choose an actions list $A$ (Choose all or drop some?)
// Compute all possible combinations of actions present in $A$ to make a N-step strategy set $A'$ {{ $a'_t, ...a'_{t+N-1}$ },..., { $a'_{t+N-1}, ...a'_t$ }}

$s'_t = s_t$

for strategy in $A'$ do

    for  action in strategy

        $s'_{t+1}$, reward$'$ = EnvironmentStep($s'_t, action$)
        $s'_t = s'_{t+1}$ do
    end for
        //Calculate Damage?
        Damage = |reward$'$ − reward|
        OR Damage = sum of cumulative rewards

        if $\Delta_1 <$ Damage $< \Delta_2$ then

        attack = True

        // store { $a'_{t+N-1}, ...a'_t$ } in dictionary against its Damage

        Dict["Damage"] = { $a'_{t+N-1}, ...a'_t$ }

        if earlyStopping == True: return Dict, attack end
else attack = False return attack
end

end
return Dict, attack

// Choose strategy { $a'_t, ...a'_{t+N-1}$ } from Dict having damage closer to $\Delta_1$
// Add targeted perturbations to take the actions specified in chosen strategy?
```

3) How these two questions related? How to Attack? and When to Attack?: There can be a relation between two questions as different situations might need different kinds of perturbation. For example, when a rapid attack is needed, we add R+FGSM perturbation and whenever there is a need of 100 % success, we add C&W perturbation or multi step R+FGSM perturbation to the observation. Therefore, we can devise a metric called Urgency. If urgency is too low, we can craft C&W perturbation but if the urgency is high, we can craft R+FGSM perturbation. This can be our second kind of attack where adversary is clever enough to know which perturbation he needs to add. The relation between two questions can also be studied from the past experience of adversary stored in memory that where certain kind of perturbation went wrong and where it went right.

describe: (1) the definition of attack vectors to be considered in this work; (2) how these attacks are modeled/considered in this work; and (3) how these attacks can be handled – when we develop a defense mechanism

VI. Experimental Setup

A. Metrics

Following are the metrics used to compare our attacks with existing attacks:

- Conventional Success Rate (CSR): We report success if the attack produced an adversarial example to take an action which reduces reward. Failure indicates the case where the attack was entirely unable to succeed or DRL agent takes the action that maximizes the reward. We show that our success rate is better as compared to other fast perturbation generation methods in DRL setting. We define success rate as the ratio of the total number of successes to total number of attempts. An attempt is defined as an attack on one time step. We call this a conventional success rate (CSR).

$$\text{CSR} = \frac{\#\text{Perturbation Reduced Reward (Successes)}}{\#\text{Steps attacked (Attempts)}}$$

- Stable Success: How to quantify stable success?
- Defender aware success rate (DASR): We introduce a novel metric called Defender aware success rate. If our attack is undetectable by the defender and yet able to reduce the reward, then we define this as actual success. Our defender keeps check on anomalous behaviors of DRL agent.

$$\text{DASR} = \frac{\#\text{Perturbation Reduced Reward and Undetectable}}{\#\text{Steps attacked (Attempts)}}$$

- Average Running time (ART): It is the average time required to generate one perturbation or a perturbed image. We compare our attack with the runtimes of existing attacks to show that our attack is faster. We calculated average running time in this way:

$$\text{ART} = \frac{\text{Time to compute all adversarial states}}{\text{Total steps attacked}}$$

- Average Reward: We define average reward as the average reward of all episodes. We compare the average reward reduced by our attack with existing attacks against state of the art defenses. We show that our attack will reduce more average reward, hence more robust against such defenses.
- Targeted Action Perturbation Rate (TAPER): TAPER is the probability at which certain perturbation generation method can produce the perturbation, which can lead to the targeted action. For example, in pong, we want to take a targeted action ’up’ at a given point, TAPER is the probability that certain perturbation can lead to targeted action.

$$\text{TAPER} = \frac{\#\text{Perturbation produced targeted action}}{\#\text{Steps attacked (Attempts)}}$$

B. Comparing Schemes

if you want to compare the strength of existing attacks with your developed attack algorithm, then what other attack algorithms you will consider? same for the defense mechanism.

We are using the following FGSM and C&W based attack schemes to compare our attack:

- Huang et al. [7] presented an attack based on FGSM. This attack serves as our baseline. We show that our attack is better in number of ways: we use sophisticated FGSM which is undetected by adversarial retraining; we consider hidden undetectable and fast end goal strategy.
- Sun et al. [20] presented an end goal based attack, in which C&W perturbations were used. We will show that our attack is faster and has comparable success.

C. Parameters

Following are the parameters we are tuning to generate our perturbations:

- Distance Metric: There are three kinds of norm constraints L_0, L_1 and L_∞ which are used in literature to constrain the perturbation in order not to be detectable. We are using all three of them to generate our perturbations.
- Epsilon: This defines the size of perturbation. It should not be very small or very large. It should be optimal.
- Number of steps in an episode: We need to make the steps in an episode as low as possible to conduct an undetectable attack.
- Number of steps of R+FGSM: We need to make the steps as low as possible to reduce the running times and yet achieve high success rate.
- Alpha: This defines the size of the step R+FGSM takes. It is less than epsilon.

D. Defender Model

We assume that our defender has some kind of detection model in order to detect the presence of adversary as our goal is to damage the DRL agent while remaining undetectable. A basic model one could think of is an anomaly detection model to check whether there is an anomaly in the states or not. Such kinds of anomaly detection techniques are presented by Lin et al. [13] and Xiang et al. [23]. Lin et al. [13] propose

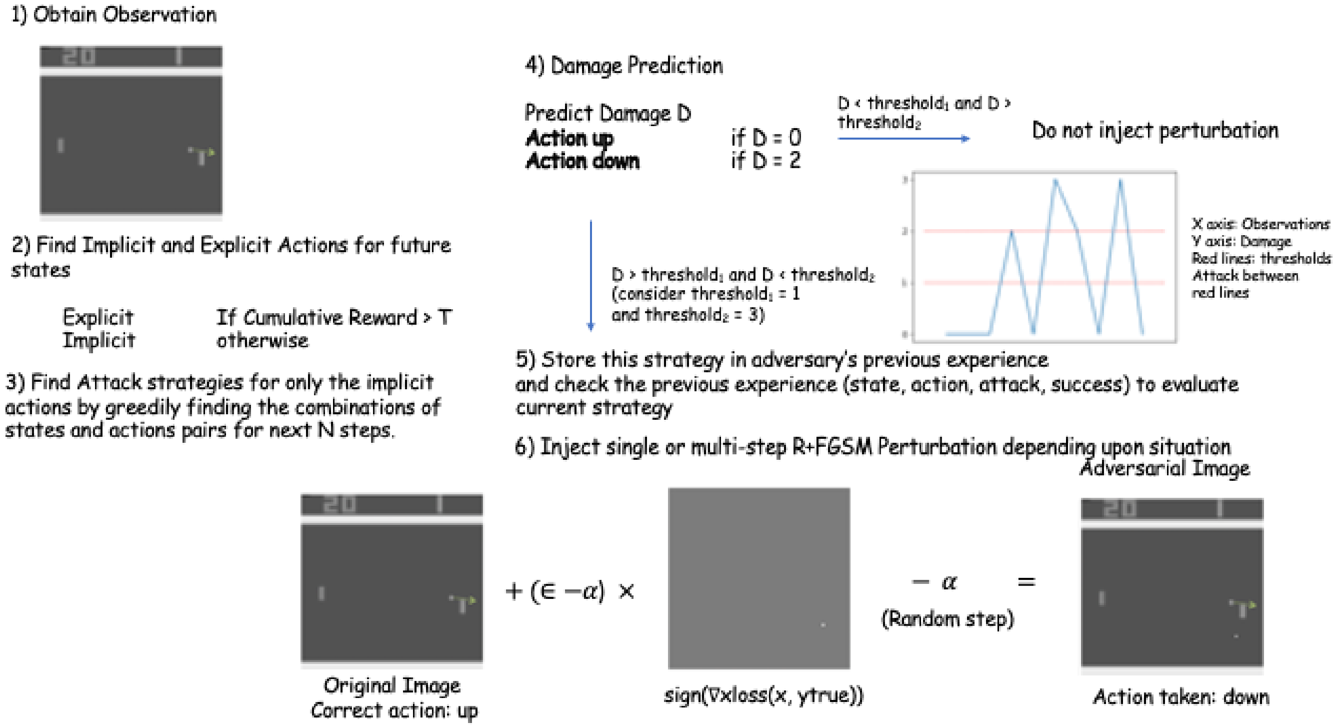
1) Obtain Observation
2) Find Implicit and Explicit Actions for future states
Explicit
If Cumulative Reward > T
Implicit
otherwise
3) Find Attack strategies for only the implicit actions by greedily finding the combinations of states and actions pairs for next N steps.
4) Damage Prediction
Predict Damage D
Action up
if D = 0
Action down
if D = 2
D < threshold1 and D > threshold2
Do not inject perturbation
D > threshold2 and D < threshold1 (consider threshold1 = 1 and threshold2 = 3)
X axis: Observations
Y axis: Damage
Red lines: thresholds
Attack between red lines
5) Store this strategy in adversary's previous experience and check the previous experience (state, action, attack, success) to evaluate current strategy
6) Inject single or multi-step R+FGSM Perturbation depending upon situation
Adversarial Image
+ (∈ −α) ×
− α
=
(Random step)
Original Image
Correct action: up
sign(∇xloss(x, ytrue))
Action taken: down

Fig. 1: 1) Obtaining the observation from Pong game and the correct action is up. 2) Now, we have to decide whether to attack or not. Finding implicit and explicit actions as we do not want to cater explicit and detectable actions. 3) Make strategies with implicit actions for future time steps 4) If any strategy falls in the window of an undetectable damage, inject perturbation. 5) Check the strategy in memory 6) In How to attack, we are adding a random step to inject a strong and robust perturbation, which resulted in suboptimal action of down

a method of protecting the DRL algorithms from adversarial attacks by leveraging an action-conditioned frame prediction module. By using this technique, they can detect the presence of adversarial attacks and make the model robust by using the predicted frame instead of the adversarial frame. Xiang et al. [23] presented a PCA based technique to train a model in order to segregate adversarial inputs from the normal ones in the robot path finding problem.

E. Evaluation against Adversarial Retraining

Adversarial retraining against FGSM based attacks was proven to be a good defense but we show that our attack is robust to such defenses.

F. Environmental Setup

discuss key design parameters setup for network, data, etc.; add a table of key design parameters, meanings, and default values used

Environment setup of some papers is shown in table II. We choose to attack on DQN and A3C agents in the Pong Atari game using R+FGSM or more sophisticated gradient based perturbation methods. We are using the implementation of Mnih et al. [17] for A3C and Mnih et al. [15] for DQN. Pong is implemented in OpenAI Gym. We choose average return or reward as our metric. We will also change epsilon and will use three of the norm constraints to observe changes in average reward. We will also consider the success rate of our attack and the number of steps required to compromise the end goal. We opted for cross entropy loss in our gradient based attacks. Most importantly, we will focus on runtimes of perturbation generation methods we are employing to attack DRL agents, which has never been the focus of previous DRL adversarial attacks.

DQN on Pong FGSM, RFGSM, CW on DQNPong, DQN Breakout Speed Analysis Test Time attack

1- How many steps we need to successfully miss the ball in Pong and Breakout? 2- Tune Damage Threshold against average return (Sun et al. also took a middle damage value but their reason was different) 3- Attack Rate vs Average Return (Attack Rate is the number of steps/total steps) (Average Return is the average reward over number of episodes) OR number of steps vs Average Return 4- Comparison graphs with previous work 5- Epsilon vs Average Return (Show 3 norms), the average return for each result reported is the average cumulative reward across ten rollouts of the target policy, without discounting rewards.

Domain Agnostic Domain Aware Attacker's DRL agent

Paper	Algorithms	Application	Metrics	Perturbations	Setting	Parameters
Sun et al. [20]	A3C, DDPG, PPO	Atari (Pong, Breakout), TORCS, Mojuco (generic for different applications)	Average Return	C&W	White Box	Number of steps, Damage Awareness Metric Delta
Lin et al. [12]	A3C, DQN	5 Atari Games (MsPacman, Pong, Seaquest, Qbert, and ChopperCommand)	Success rate, Accumulated Reward	C&W	White Box	Number of steps, Epsilon
Behzadan and Munir [3]	DQN	Pong	Success rate, Average Reward Per Epoch	FGSM, JSMA	Black Box	Number of Observations, Epochs
Huang et al. [7]	A3C, TRPO, DQN	Chopper Command, Pong, Seaquest, and Space Invaders	Average Return	FGSM	Both blackbox and white box	Epsilon, 3 Norm constraints
Pattanaik et al. [18]	DDPG, DDQN	MuJoCo, Cart Pole, Mountain Car, Hopper, Half Cheetah	Total return per episode	GB (Gradient based, similar to FGSM)	White Box	Seeds, Friction, Length of Pole etc.
Kos and Song [9]	A3C	Pong	Reward	FGSM, Random Noise	White Box	Epsilon, Noise amount, Value function

TABLE I: Environment Setup of papers

Perturbation Method	Success Rate	Average Time (seconds)	Targeted Action Rate
CW	1	0.83	0.97
7-step RFGSM	1	0.042	0.98
6-step RFGSM	1	0.036	0.97
5-step RFGSM	1	0.032	0.96
4-step RFGSM	1	0.024	0.90
3-step RFGSM	1	0.018	0.82
2-step RFGSM	0.98	0.011	0.63
1-step RFGSM	0.91	0.006	0.31
FGSM	0.85	0.006	0.57

TABLE II: Comparison between different Perturbation Methods based on success rate, average one perturbation generation time and targeted action rate

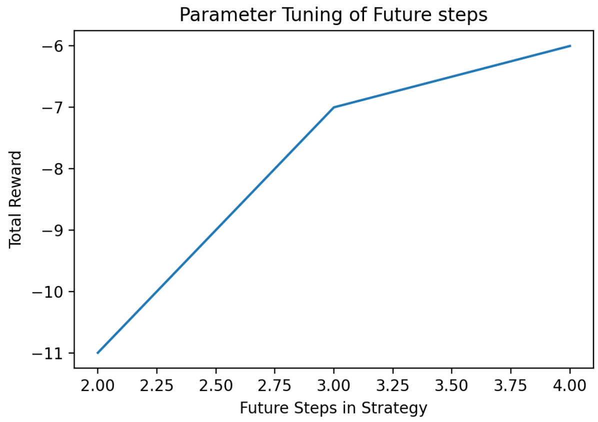

Fig. 2: Future Steps Affect on Reward (Domain Agnostic)

VII. Results & Analysis

VIII. Conclusion & Future Work

References

[1] M. Alzantot, B. Balaji, and M. Srivastava, "Did you hear that? adversarial examples against automatic speech recognition," *arXiv preprint arXiv:1801.00554*, 2018.

[2] S. Amarjyoti, "Deep reinforcement learning for robotic manipulation-the state of the art," *arXiv preprint arXiv:1701.08878*, 2017.

[3] V. Behzadan and A. Munir, "Vulnerability of deep reinforcement learning to policy induction attacks," *CoRR*, vol. abs/1701.04143, 2017. [Online]. Available: http://arxiv.org/abs/1701.04143

[4] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*. IEEE, 2017, pp. 39–57.

[5] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[6] ——, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[7] S. H. Huang, N. Papernot, I. J. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," *CoRR*, vol. abs/1702.02284, 2017. [Online]. Available: http://arxiv.org/abs/1702.02284

[8] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[9] J. Kos and D. Song, "Delving into adversarial attacks on deep policies," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017. [Online]. Available: https://openreview.net/forum?id=BJcib5mFe

[10] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint*

arXiv:1611.01236, 2016.

[11] A. Kurakin, I. Goodfellow, S. Bengio *et al.*, "Adversarial examples in the physical world," 2016.

[12] Y.-C. Lin, Z.-W. Hong, Y.-H. Liao, M.-L. Shih, M.-Y. Liu, and M. Sun, "Tactics of adversarial attack on deep reinforcement learning agents," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017, pp. 3756–3762. [Online]. Available: https://doi.org/10.24963/ijcai.2017/525

[13] Y.-C. Lin, M.-Y. Liu, M. Sun, and J.-B. Huang, "Detecting adversarial attacks on neural network policies with visual foresight," *arXiv preprint arXiv:1710.00814*, 2017.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[16] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[17] ——, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[18] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary, "Robust deep reinforcement learning with adversarial attacks," *arXiv preprint arXiv:1712.03632*, 2017.

[19] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[20] J. Sun, T. Zhang, X. Xie, L. Ma, Y. Zheng, K. Chen, and Y. Liu, "Stealthy and efficient adversarial attacks against deep reinforcement learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, pp. 5883–5891, Apr. 2020. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/6047

[21] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[22] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[23] Y. Xiang, W. Niu, J. Liu, T. Chen, and Z. Han, "A pca-based model to predict adversarial examples on q-learning of path finding," in *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*. IEEE, 2018, pp. 773–780.